ERDC MSRC/PET TR/00-30

HLA Integration for HPC Applications Applied to CMS

by

Wojtek Furmanski
David Bernholdt
Geoffrey Fox

28 June 2000

07h0112000

HLA Integration for HPC Applications Applied to CMS

ERDC PET FMS Year 4 Focused Project Technical Report

Wojtek Furmanski, David Bernholdt, Geoffrey Fox *(contact person at fox@csit.fsu.edu)*
NPAC, Syracuse University

Syracuse, NY, June 2000.

Introduction We present here our first results from the genuine metacomputing demonstrations (including four geographically distributed and collaborating labs) of our HLA based framework for integrating high performance distributed modeling and simulation applications. Our approach presented here explores synergies among and integrates distributed object standards emerging from industry (CORBA), Web (Java, XML) and the DoD (HLA). More specifically, we developed a 3-tier WebHLA environment that offers standards based plug-and-play support both for the back-end HPC simulation modules and for the front-end Web/Commodity interfaces. In this report, we overview the DoD Modeling and Simulation domain from the perspective of HPC, we summarize the High Level Architecture (HLA) standard, we outline our WebHLA environment and we illustrate its use for building a metacomputing level ModSAF based battlefield simulation that involves large scale minefields (of order of million mines), simulated by the Parallel CMS (Comprehensive Mine Simulator) module running on Origin2000.

DoD Modeling and Simulation (M&S) Modeling and Simulation is a major computationally intense mission-critical domain of DoD computing. It addresses a broad range of application areas ranging from weapon engineering to multi-player training to campaign analysis, and it includes a spectrum of granularity and fidelity levels ranging from close combat to entity level to force-on-force simulations. Being naturally modular in terms of distributed simulation entities, DoD Modeling and Simulation always acted as a driving force for new distributed computing and network technologies. Based on lessons learned from SIMNET, the first generation standards emerged such as DIS (Distributed Interactive Simulation) for real-time simulations or ALSP (Aggregate Level Simulation Protocol) for logical-time simulations. Several large scale joint enterprises now address various aspects of the broad field of M&S, including JSIMS (Joint Simulation System) for training simulations, JMASS (Joint Modeling and Simulation System) for engineering simulations and JWARS (Joint Warfare Systems) for campaign level analytical simulations. These large scale efforts were accompanied by numerous smaller scale modeling and simulation activities in many DoD labs so that the whole field was significantly fragmented until recently. New mechanisms for simulation interoperability are being developed and enforced recently by DMSO (Defense Modeling and Simulation Office) in terms of the HLA (High Level Architecture) based federation framework discussed below.

Forces Modeling and Simulation (FMS) One relatively small but special sector on the large DoD Modeling and Simulation landscape called FMS (Forces Modeling and Simulation) is focused on large scale simulations that require HPC support. Most other CTAs within the DoD HPC Modernization Program such as CFD, CSM, CEA, etc., are based on traditional data parallel time-stepped HPC simulation technologies, whereas FMS represents a special domain of object-oriented event-driven task parallel HPC simulations. Parallel and distributed event-driven simulations (PDES) are often classified according to the "real-time" (or "as-fast-as-possible") or "logical-time" management scheme. The former, typically used for real-time battlefield simulations, e.g., for training purposes were usually based on DIS protocol. In such simulations, all active objects (vehicles, troops, weapons etc.) broadcast periodically their entity state PDUs (Protocol Data Units), informing all other players on their positions and internal state. Based on

received PDUs, all entities update their states "as-fast-as-possible" and the resulting simulation advances in "real-time". In the logical time management mode, simulation objects generate events and schedule them for execution at some future time instances. For example, when a missile is fired, its space-time collision point is pre-computed and the corresponding "target hit" event is constructed and put into the time-ordered queue for future execution. Simulation time advances in discrete irregular steps, given by the timestamps of the subsequent events in the queue.

Both time management regimes are being addressed by FMS projects. In the logical time domain, the dominant PDES technology is based on the SPEEDES (Synchronous Parallel Environment for Emulation and Distributed Events Simulation) system by Metron Corporation. SPEEDES uses an optimistic rollbackable parallel time management scheme based on a variant of the Time Warp algorithm developed by NASA/JPL in late '80s. In the real-time domain, the DIS based battlefield simulations map naturally on networks of workstations and hence the use of MPPs was rather limited in this area. However, there are some specific DIS simulation problems that require HPC. One of such challenges, raised recently by Ft. Belvoir, VA, addressed support for entity level battlefield simulation in large minefields (of million or more mines) that are required by modern warfare models. We will discuss this Comprehensive Mine Simulator (CMS) application and our support for Parallel CMS in another Year 4 ERDC technical report [4] (see also the CRPC book chapter [3]). In this document, we focus on the WebHLA integration environment that was used to support Metacomputing CMS runs. We first summarize the current status in the area of simulation interoperability, represented by the HLA federation framework, and we then describe our WebHLA architecture and the Metacomputing CMS application.

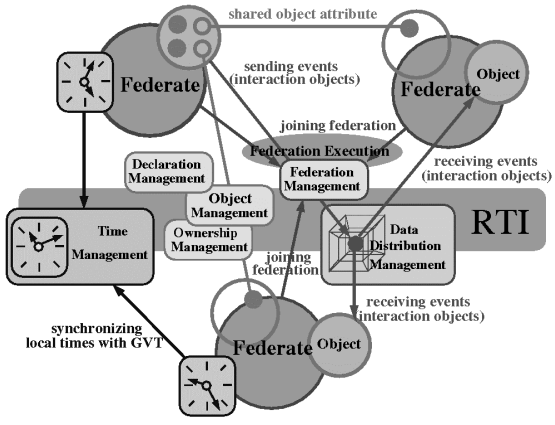

Fig. 1: *Architecture of the Run-Time Infrastructure (RTI) software bus of the High Level Architecture (HLA) - circles represent entities (such as federates, objects, attributes), rectangles represent services.*

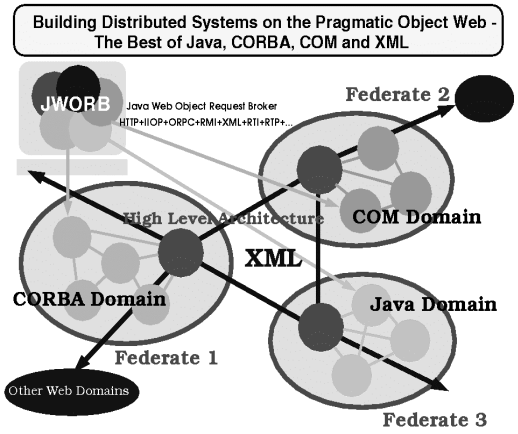

Fig. 2: *Pragmatic Object Web architecture - fine grain distributed objects of CORBA, Java and COM interoperate as coarse grain HLA federates linked via XML messages.*

High Level Architecture (HLA) HLA is a language-independent object-based distributed software architecture for simulation reusability and interoperability that is now being enforced DoD-wide across all individual M&S programs, systems and simulation paradigms, including both real-time (DIS) and logical time (event-driven) management models. HLA views distributed simulation as a *federation* of coarse grain opaque semi-autonomous entities called *federates* that govern locally and independently their simulation objects and that conform strictly to some global federation rules, specifying the information exchange policy across the federation. The associated Run-Time Infrastructure (RTI) offers the software bus services available to the HLA-compliant federates and including Federation, Object, Declaration, Ownership, Time and Data Distribution Management. We illustrate the overall organization of RTI in Fig 1. Federates (large circles)

maintain their simulation objects (medium circles) given by attribute sets (small circles) and they interact via RTI services (rounded rectangles) managed by the RTI bus (central elongated rectangle). Both local simulation and global federation objects conform to a simple attribute-value based entity format specified by the Object Model Template (OMT) and are suitably grouped and maintained by the RTI as SOMs (Simulation Object Models) or FOMs (Federation Object Models). Federates can join or leave federation using Federation Management, they create their objects and register them with the RTI using Object Management, they can publish and/or subscribe some of their objects or their selected attributes for sharing using the Declaration Management, they can negotiate update rights for shared objects using Ownership Management, they can evolve their objects in time and they can synchronize their local simulation clocks with the federation time using Time Management, and they can build dynamic multi-dimensional routing channels for optimized multicast delivery of discrete communication events called interaction objects using Data Distribution Management.

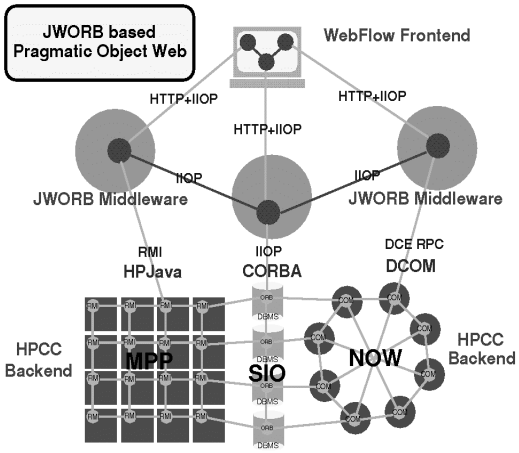

Fig. 3: *Overall architecture of the multi-protocol JWORB server - front-end browsers (orblets) connect via HTTP (IIOP), middleware is IIOP based, legacy backends are linked via dedicated protocols.*

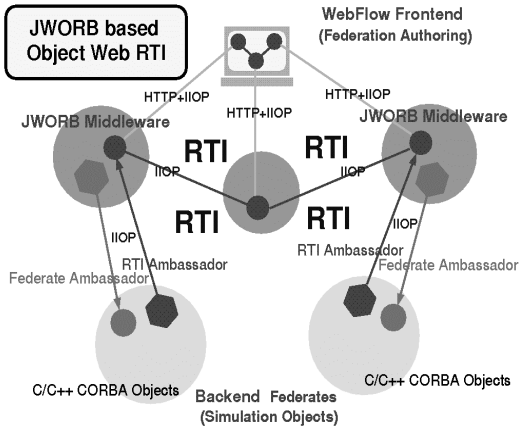

Fig. 4: *Overall architecture of OWRTI, packaged as JWORB facility. RtiCap library is employed to link C++ simulation backends via RTI in terms of RTI Ambassador and Federate Ambassador proxies.*

WebHLA DMSO's main emphasis so far was on supporting reusability of and HLA-enabled interoperability among diverse existing legacy codes rather than on providing HLA based software engineering support for new simulations that would utilize the latest Web/Commodity technologies of Java, CORBA and XML. We recently proposed to fill this gap in our WebHLA [1][2] framework that offers open implementation of HLA in terms of a suite of emergent object standards for the Web based distributed computing - we call it *Pragmatic Object Web* - that integrate Java, CORBA, COM and XML (see Fig. 2). WebHLA is an interactive 3-tier environment including: a) DMSO HLA architecture and our JWORB based Object Web RTI implementation in the *middleware*; b) Web/Commodity *front-ends* such as Web browsers or Microsoft Windows; and c) Customer and application specific *back-end* technologies ranging from legacy systems such as relational databases to HPC modeling and simulation modules. Below, we outline both the core components of WebHLA such as JWORB and OWRTI and a suite of tools and plug-and-play federates developed so far and including RtiCap, JDIS, PDUDB and SimVis.

JWORB (Java Web Object Request Broker) is a multi-protocol network server written in Java (see Fig. 3). Currently, JWORB supports HTTP and IIOP protocols, i.e., it can act as a Web server and

as a CORBA broker or server. In progress is support for the DCE RPC protocol which will provide COM server capabilities. JWORB recognizes a particular protocol based on the anchor/magic number of the current network packet and it invokes a suitable handler. JWORB is a useful middleware technology for integrating and efficiently aggregating competing distributed object technologies and the associated network protocols of CORBA, Java, COM and XML.

OWRTI (Object Web RTI) is an implementation of DMSO RTI 1.3 written in Java on top of the JWORB middleware, i.e., packaged as a JWORB CORBA service (see Fig 4). In OWRTI, each of the RTI management services shown in Fig. 1 is implemented as an independent CORBA object. Other CORBA objects in the system include: *RTIKernel* which acts as a core top level manager, *FederationExecution* which represents a federation instance, *RTIAmbassador* which acts as a client side proxy of the RTI bus, and *FederateAmbassador* which acts as the RTI side proxy of a federate.

RtiCap library provides RTI C++ programming interface, packaged as a CORBA service that offers access to Java based OWRTI from C++ federates. RtiCap glue library uses public domain OmniORB2 as a C++ Object Request Broker. RTI Ambassador glue/proxy object forwards all C++ client method calls to its Java/CORBA peer and the Federate Ambassador object forwards all received callbacks to its C++ peer. Versions of RtiCap library are running on Windows NT, IRIX and SunOS platforms.

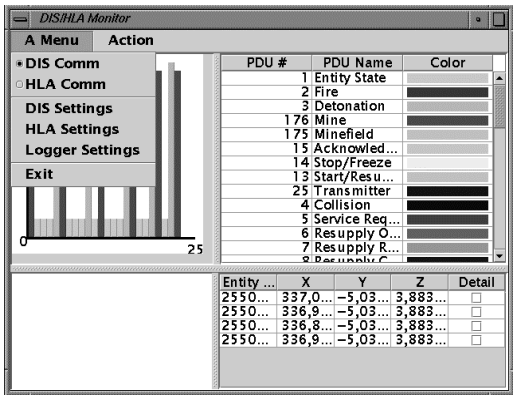

Fig. 5: *A sample screen of the JDIS and PDUDB control monitor window, illustrating the dynamic display of the PDU flow and various protocol and I/O modes (DIS vs HLA, runtime vs playback).*

Fig. 6: *A sample screen of SimVis, used to visualize a battlefield (including tanks propagating through a terrain with deployed minefield) associated with Parallel CMS + ModSAF simulation.*

JDIS To link DIS based legacy simulation systems such as ModSAF (Modular Semi-Automated Forces) with HLA federations, a bridge node is required to transform between different event models used in both frameworks: DIS PDUs (Protocol Data Units) and HLA Interactions. We constructed such a bridge called JDIS in Java, starting from a public domain DIS Java parser and completing it to support all PDUs required by the ModSAF system. JDIS can also write / read PDUs from a file or a database and hence it can be used to log and playback sequences of simulation events. In order to facilitate the transmission of PDUs and their persistent storage, we adopted XML as a uniform wire format and we constructed suitable PDU-XML converters.

PDUDB Playing the real scenario over and over again for testing and analysis is a time consuming and tedious effort. A database of the equivalent PDU stream is often needed for

selectively playing back segments of a once recorded scenario. We constructed and packaged as a WebHLA federate such a PDU database, using Microsoft's Access for storage, Java servlets for loading and retrieving the data, and JDBC for servlet-database communication. The PDU logger servlet receives its input via HTTP POST message in the form of XML-encoded PDU sequences. Such input stream is decoded, converted to SQL and stored in the database using JDBC. The playback is done using another servlet that sends the PDUs generated from the database as a result of a query. A common visual front-end for JDIS and PDUDB federates is shown in Fig. 5. It supports runtime display of the PDU flow, and it offers several controls and utilities, including: a) switches between DIS, HLA and various I/O (file, database) modes; b) frequency calibration for a PDU stream generated from file or database; c) PDU probe and sequence generators; and d) simple analysis tools such as statistical filters or performance benchmarks that can be performed on accumulated PDU sequences.

SimVis Using Microsoft Direct3D technology, we constructed a real-time battlefield visualizer, SimVis (see Fig. 6) that can operate both in the DIS and HLA modes. SimVis is an NT application written in Visual C++ that extracts the battlefield information from the event stream, including state (e.g. velocity) of vehicles in the terrain, position and state of mines and minefields, explosions that occur e.g. when vehicles move over and activate mines, etc. The renderer performs the real-time visualization of the extracted information, using the ModSAF terrain database, a suite of geometry objects and animation sets for typical battlefield entities such as armored vehicles (tanks) and visual events such as explosions. We developed these objects using the 3D Studio MAX authoring system and we imported them into the DirectX/Direct3D runtime environment.

Example WebHLA Application: Parallel/Metacomputing CMS Having outlined our WebHLA framework we illustrate now its application in a particular FMS project conducted by NPAC that developed Parallel and Metacomputing CMS based on the CMS simulator from Ft. Belvoir. This effort included converting the CMS system from the DIS to HLA framework, constructing scalable Parallel CMS federate for Origin2000 and linking it with ModSAF vehicle simulator and other utility federates towards a Metacomputing CMS federation. In the following, we review the original CMS system and we describe our WebHLA based Metacomputing CMS demonstration.

Comprehensive Mine Simulator by Ft. Belvoir The Night Vision Lab at Ft. Belvoir, VA conducts R&D in the area of countermine engineering, using the advanced Comprehensive Mine Simulator (CMS) as an experimentation environment for a synthetic battlefield. Developed by the OSD sponsored Joint Countermine Advanced Concepts Technology Demonstration (JCM ACTD), CMS is state-of-the-art high fidelity minefield simulator with support for a broad range of mine categories, including conventional types such as buried pressure-fuzed mines, antitank mines and other types including off-route (side attack) and wide-area (top attack) mines. CMS organizes mines in components, given by regular arrays of mines of particular types. Minefields are represented as heterogeneous collections of such homogenous components. CMS interoperates via the DIS protocol with ModSAF vehicle simulators. Mine interaction with a target is controlled by its fuse. CMS supports several fuze types, including full width, track width fuzes, off-route fuzes and others. CMS mines can also interact with countermine systems, including both mechanical and explosive countermeasures and detectors

The relevance of HPC for the CMS system stems from the fact that modern warfare can require a million or more of mines to be present on the battlefield, such as in the Korean Demilitarized Zone or the Gulf War. The simulation of such battlefield areas requires HPC support. As part of the PET FMS project, Syracuse University analyzed the CMS code, ported the system to the

Origin2000 shared memory MPP and repackaged it as an HLA federate. A more detailed description of our Parallel CMS federate and the performance results can be found in another Year 4 PET FMS ERDC Technical Report [4] (see also the CRPC book chapter [3]).

Metacomputing CMS The timing results for the Parallel CMS module described in [4] were obtained during Parallel CMS runs within a WebHLA based HPDC environment that span three geographically distributed laboratories and utilized most of the WebHLA tools and federates discussed above. The overall configuration of such initial Metacomputing CMS environment is shown in Figs. 7 and 8. ModSAF, JDIS and SimVis modules were typically running on a workstation cluster at NPAC in Syracuse University. JWORB/OWRTI based Federation Manager (marked as FE = Federation Execution in Fig. 7 and as RTI in Fig. 8) was typically running on Origin2000 at ERDC in Vicksburg, MS. Parallel CMS federate was typically running on Origin2000 at NRL in Washington, DC.

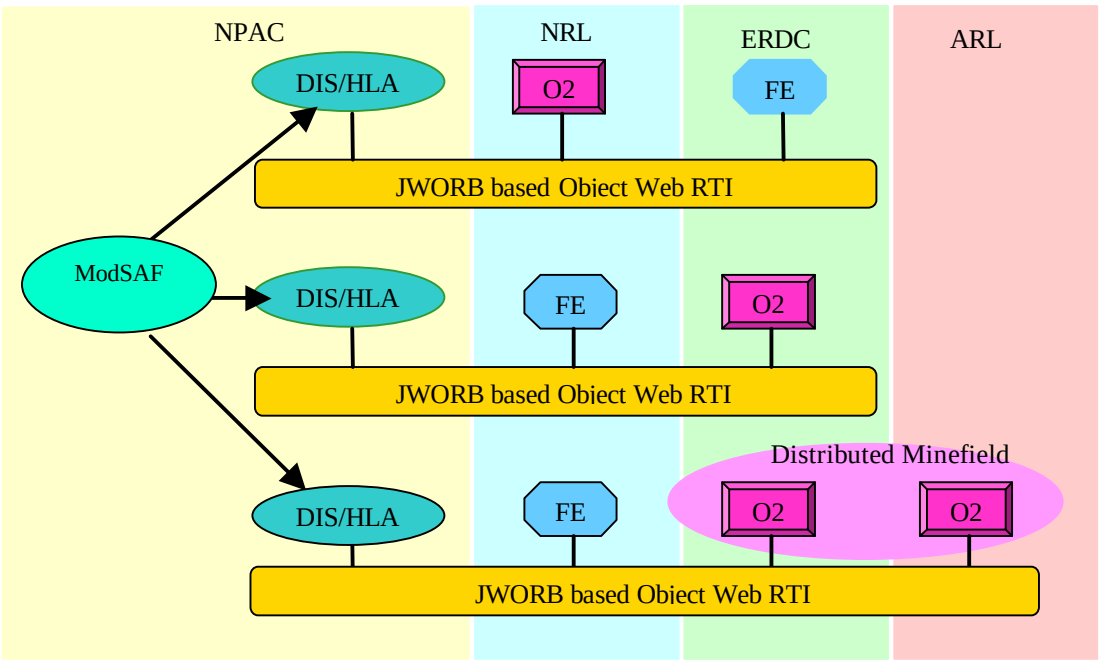

Fig. 7 *Typical configurations of our WebHLA based Metacomputing CMS used to measure scalability of Parallel CMS, running for one million mines on NRL Origin2000 (top federation) and using ERDC facilities for federation execution management (FE). We were also collecting timing results using ERDC Origin2000 (middle federation) and NRL for federation management. Finally, we also performend initial runs in the distributed minefield mode with two halves of a large minefield running concurrently on ERDC and NRL or*

Large MISER runs at NRL need to be scheduled in a batch mode and are activated at unpredictable times, often in the middle of the night. This created some logistics problems since ModSAF is a GUI based legacy application that needs to be started by a human pressing the button. To bypass the need for a human operator to continuously monitor the MISER batch queue and to start ModSAF manually, we constructed a log of a typical simulation scenario with some 30 vehicles and we played it repetitively from the database using our PDUDB federate. The only program running continuously (at ERDC) was the JWORB/OWRTI based Federation Manager. After the Parallel CMS was started by MISER at NRL, it joined a distributed federation (managed at ERDC) and automatically activated the PDUDB playback server at NPAC that

started to stream vehicle PDUs to JDIS which in turn converted them to HLA interactions and sent (via RTI located at ERDC) to the Parallel CMS federate at NRL. Each such event, received by node 0 of Parallel CMS was multicast via shared memory to all nodes of the simulation run and used there by the node CMS programs to update the internal states of simulation vehicles. The inner loop of each node CMS program was continuously tracking all mines scattered into this node against all vehicles in search of possible explosions (see refs [3] and [4] for description of the Parallel CMS internals).

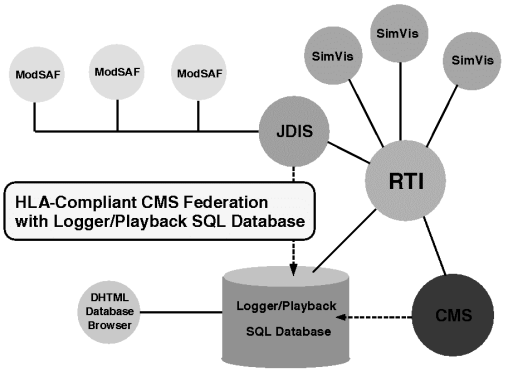

Fig. 8*: A WebHLA environment that supports Parallel CMS experiments and includes: ModSAF vehicles, SimVis front-ends, JDIS bridge between DIS and HLA domains, event logger and playback database, Parallel CMS and RTI Federation Mgr*

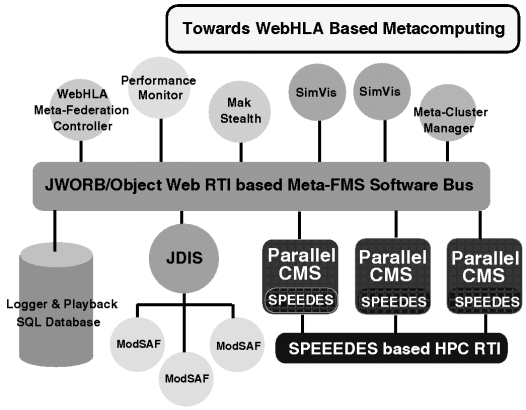

Fig. 9*: Planned Metacomputing CMS with WebHLA based distributed management similar as in Fig. 8 and with SPEEDES based HPDC support for large scale geographically distributed minefields.*

Next Steps Having constructed a fully scalable Parallel CMS federate and having established a robust Metacomputing CMS experimentation environment, we intend now to proceed with the next set of experiments towards wide area distributed large scale FMS simulations, using CMS as the application focus and testbed.

In the first such experiment, we intent to distribute large minefields of millions of mines over several Origin2000 machines in various DoD labs using domain decomposition, followed by the scattered decomposition of each minefield domain over the nodes of a local parallel system. So far, we obtained first results for Distributed Parallel CMS with 30K mine domains of a 60K mine minefield distributed over Origins in ERDC and ARL facilities. Parallel CMS runs for minefields larger than 30K mines need to be executed via local batch queues and hence a robust metacomputing operation would require global synchronization between such schedulers which is the subject of one of our proposed Year 5 tasks.

Our other planned effort includes support for parallel object-oriented tools and authoring environments - we propose to accomplish it by integrating our previous WebFlow and WebHLA efforts with the new industry standards for object analysis and design such as UML (Uniform Modeling Language).

In another proposed experiment, we intend to replace our simple SPEEDES micro-kernel discussed above by the full SPEEDES simulation kernel as illustrated in Fig 9. This way, we will be able to offer optimized communication between individual MPPs using the SPEEDES based HPC RTI under development by Metron, and to convert the legacy CMS code to a well-organized

programming model of SPEEDES. One ofour tasks within the PET FMS program is to provide Web based SPEEDES training for the DoD users and we view our WebHLA metacomputing environment, described in this report, as a useful training framework to be employed for this task in the context of Metacomputing CMS as a trial large scale FMS application.

References

1. Geoffrey C. Fox, Ph. D., Wojtek Furmanski, Ph. D., Ganesh Krishnamurthy, Hasan T. Ozdemir, Zeynep Odcikin-Ozdemir, Tom A. Pulikal, Krishnan Rangarajan, Ankur Sood, "Using WebHLA to Integrate HPC FMS Modules with Web/Commodity based Distributed Object Technologies of CORBA, Java, COM and XML," In Proceedings of the Advanced Simulation Technologies Conference ASTC 99, San Diego, April 99.

2. G. Fox, W. Furmanski, G. Krishnamurthy, H. Ozdemir, Z. Ozdemir, T.Pulikal, K. Rangarajan and A. Sood, "WebHLA as Integration Platform for FMS and other Metacomputing Application Domains," In Proceedings of the DoD HPC Users Group Conference, Monterey, CA, June 8-15, 1999.

3. CRPC Book Chapter, Morgan-Kaufmann (in progress): WebHLA based Metacomputing Environment for Forces Modeling and Simulation.

4. Wojtek Furmanski, David Bernholdt, Geoffrey Fox, "Enforcing Scalability of Parallel Comprehensive Mine Simulator (CMS)," ERDC MSRC PET Technical Report No. TR00-29. Vicksburg, MS, June 2000.